NONSMOOTH OPTIMIZATION USING UNCONTROLLED INEXACT INFORMATION

JÉRÔME MALICK*, WELINGTON OLIVEIRA†, AND SOFIA ZAOURAR‡

Abstract. We consider convex nonsmooth optimization problems whose objective function is known through a (fine) oracle together with some additional (cheap but poor) information – formalized as a second coarse oracle wih uncontrolled inexactness. It is the case when the objective function is itself the output of an optimization solver, using a branch-and-bound procedure, or decomposing the problem into parallel subproblems. Minimizing such objective function can be done by any bundle algorithm using only the (fine) oracle. In this paper, we propose a general scheme to incorporate the available coarse information into bundle-type methods in view of generating better iterates and then accelerating the algorithms. We study two pratical versions of the scheme: a (simple) inexact Kelley method, and a (sophisticated) level bundle method. We prove that these methods are convergent, and we present numerical illustrations showing they speed up resolution.

Key words. Nonsmooth optimization, bundle methods, inexact oracle, Lagrangian relaxation, Benders decomposition, convergence analysis

AMS subject classifications. 65K05, 49J52, 49M27, 90C15, 90C25, 90C27

1. Introduction: context, problematic and generic scheme.

1.1. Nonsmooth minimization with an (inexact) oracle. We consider nonsmooth optimization problems of the form

$$f_* := \inf_{x\in X} f(x)\,, \tag{1.1}$$

with a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$; and a polyedral set $X \subset \mathbb{R}^n$, and we assume that the infimum is finite ($f_* > -\infty$). Typically, nonsmoothness of f comes after a maximizaton, i.e. when f itself is the result of an inner optimization subproblem

$$f(x) = \sup_{u\in U} h(u,x) \tag{1.2}$$

where the functions $h(u,\cdot)$ are convex for each u lying in a set U. Such nonsmooth objective functions appear for example in Lagrangian relaxation (see e.g. [Lem01]), in stochastic optimization with recourse (see e.g. [SDR09]), or in Benders decomposition (see e.g. [Geo72]).

For a fixed accuracy $\eta \geq 0$, a so-called η-oracle of f provides, for a point $x \in X$ as an input, an approximate value of the function and an approximate subgradient

$$\left\{\begin{array}{lll} f_x \in \mathbb{R} & \text{such that} & f(x) - \eta \leq f_x \leq f(x), \\ g_x \in \mathbb{R}^n & \text{such that} & f_x + g_x^\top(\cdot - x) \leq f(\cdot). \end{array}\right. \tag{1.3}$$

Note that g_x is an η-subgradient: combining the two inequalities of (1.3) gives $f(x) - \eta + g_x^\top(\cdot - x) \leq f(\cdot)$, that is, $g_x \in \partial_\eta f(x)$.

If the oracle error is null ($\eta = 0$), the oracle returns the exact value $f_x = f(x)$ and a subgradient $g_x \in \partial f(x)$. For some problems, as in large-scale stochastic optimization or in combinatiorial optimization, computing exact information on f is expensive, or even out-of-reach, whereas computing some inexact information ($\eta > 0$) is still possible. For example,

*CNRS, Lab. J. Kuntzmann, Grenoble, France (jerome.malick@inria.fr)
†IMPA, Rio de Janeiro, Brazil (wlo@impa.br)
‡INRIA, UJF, Grenoble, France (sofia.zaourar@inria.fr)

when f is given by (1.2), any $\bar{u} \in U$ gives an inexact value and an approximate subgradient of f at a given $x \in X$. Indeed, convexity of $h(u,\cdot)$ yields

$$h(\bar{u},x) + g^\top(z-x) \leq h(\bar{u},z) \leq f(z), \quad \text{for any } g \in \partial_x h(\bar{u},x).$$

So we have inexact information on f by taking

$$f_x = h(\bar{u},x) \quad \text{and} \quad g_x = g \in \partial_x h(\bar{u},x). \tag{1.4}$$

In this case, an η-oracle is an optimization method maximizing $h(\cdot,x)$ over U up to the tolerance η, that is, computing $\bar{u} \in U$ that satisfies

$$f(x) - \eta \leq h(\bar{u},x) \leq f(x), \tag{1.5}$$

so that (1.4) gives the η-information (1.3).

Among the nonsmooth optimization methods to solve problems (1.1) with f known by an oracle (1.3), are the bundle-type methods: the Kelley method [Kel60, HUL93], proximal bundle methods [HUL93], level bundle methods [LNN95], and generalized bundle methods [Fra02]. Initially devellopped for exact oracle ($\eta = 0$), these methods have been extended to handle inexact oracles ($\eta > 0$) and to solve (1.1) up to an accuracy of η. Complete convergence analysis of these methods exists; roughly speaking we can show that, under some assumptions, the iterates x_k are an η-minimizing sequence

$$f_* \leq \liminf f(x_k) \leq f_* + \eta. \tag{1.6}$$

We refer to [Hin01] and [Sol03] for first articles, [ZPR00] for an inexact version of the Kelley method, [Fáb00] for an inexact level method, [Kiw06] and [OSL13] for an inexact proximal bundle method, and [OS13] for an inexact level method with vanishing errors.

1.2. Inexact oracle... and more. For some optimization problems as above with a convex function f and an η-oracle, there is in fact additional uncontrolled information, which is already available or cheap to get.

A typical example is in combinatorial optimization when f has the form (1.2), with a discrete set U and with a Lagrangian functions h (see e.g. [Lem01]). In this case, exact or approximate resolution shemes can create "good" feasible points $\bar{u} \in U$, that give, in turn, information on f by (1.4) – but with uncontrolled accuracy, so that this information cannot be used for an oracle with fixed accuracy η. For instance, suppose that (1.2) is solved by a branch-and-bound algorithm; then several feasible solutions are generated during the exploration the branch-and-bound tree, but only the final one, the optimal solution, is used by the oracle to generate (1.3). The (uncontrolled) information (1.4) produced by the intermediate feasible solutions is not used, whereas it is available for free and possibly fine (since nearly optimal solutions are usually obtained soon in the branch-and-bound process).

Another example is in stochastic optimization, for two-stage stochastic linear problems for instance (see e.g. [SDR09]). In this case, the function has a form (1.2) with separable terms corresponding to a linear term plus linear maximization subproblems

$$f(x) = c^\top x + \sum_{i=1}^{N} p_i f_i(x) \qquad \text{with} \quad f_i(x) = \sup_{W^\top u \leq q} (h_i - Tx)^\top u, \tag{1.7}$$

for given N, p_i, h_i, T, W and q (we will come back in more details in Section 4). For stochastic problems with such objectives, computing exact information on f requires to solve the N linear optimization subproblems, which can be costly when N is large. Solving only a fraction

of these subproblems (say $N/5$) still gives inexact information of f. Indeed if we compute $\bar{u}_i$ an optimal solution giving $f_i(x)$, then we can also use it to under-approximate other terms $f_j(x)$ (since the feasible sets are the same, we have $(h_j - Tx)^\top \bar{u}_i \le f_j(x)$). Thus, for a given fraction of solved problems, we have inexact information but with a unknown accuracy.

In this paper, we formalize the situation where we can compute fine (inexact or not) information together with some coarse (inexact) information by assuming we have two oracles

$$\begin{array}{c}\text{an expensive, fine oracle with accuracy bounded by } \eta \ge 0\\ \text{a cheap, coarse oracle with uncontrolled accuracy}\end{array} \tag{1.8}$$

We explore how to use these two oracles to accelerate the minimization of f by (convergent) bundle methods.

1.3. About using fine and coarse bundle information together. Assume that we are at iteration k of a bundle-like method solving (1.1). We have a sequence of points $\{x_i\} \subset X$ for which we have the linearizations

$$\bar{f}_i(\cdot) := f_{x_i} + g_{x_i}^\top(\cdot - x_i) \quad \big(\le f(\cdot) \big) \tag{1.9}$$

given by the information (f_{x_i}, g_{x_i}) computed by one of the two oracles (1.8). Let us denote by J_k^{f}, respectively J_k^{c}, the indexes i for which the fine oracle, respectively the coarse one, has been called. Bundle methods (see e.g. [HUL93, LNN95]) have in common the use of the so-called cutting-plane model of f using linearizations at previous iterates

$$\check{f}_k(\cdot) := \max_{i \in J_k^{\mathrm{f}} \cup J_k^{\mathrm{c}}} \bar{f}_i(\cdot) \quad (\le f(\cdot)\,). \tag{1.10}$$

This model is used to compute the next iterate x_{k+1} by solving a problem of the type

$$\min_{x \in X} \ \check{f}_k(x) + \frac{1}{2t_k}\|x - \hat{x}_k\|^2 \qquad \text{or} \qquad \min_{x \in X:\, \check{f}_k(x) \le f_k^{\mathrm{lev}}} \ \frac{1}{2}\|x - \hat{x}_k\|^2 \tag{1.11}$$

for a current "stability center" $\hat{x}_k$, and a "prox-parameter" t_k or a "level parameter" f_k^{lev}, following the usual terminology of bundle methods [HUL93].

The model (1.10) using all the information (max on both J_k^{f} and J_k^{c}) is obviously always above the model of f that would restrict to the fine one (max on J_k^{f} only). It is thus clear that using both fine and coarse information give a more precise model, so would possibly lead to computation of better iterates. Admittedly, in practice using the complete model (1.10) rather than ignoring coarse information makes quadratic programming problems (1.11) larger and then more difficult to solve. This is partly compensated by the ever-growing performance of (specific or even general-purpose) linear-quadratic programming solvers. Anyway, this drawback does not really hold in the case of expensive oracles – which is the situation we consider in this paper. Thus, there is a clear practical interest to consider as much information as possible when solving (1.1) with bundle methods: richer information can accelerate numerical methods at a neglectable cost, so that the overall computing time is lower than using only the fine information. This will be illustrated on examples coming from stochastic optimization in the numerical experiments of Section 4.

There is nevertheless a theoretical argument against using the coarse information in the model. Up to our understanding, the convergence results of bundle methods do not extend in a straightforward way for handling general models (1.10). Proofs of convergence use indeed that iterates are computed using a cutting-plane model with "controlled" linearizations, produced by an oracle with bounded η, or vanishing $\eta \to 0$ (see e.g. [Fáb00], [Kiw06], [OSL13],

[OS13]). In other words, it is not possible to consider $J_k^c \neq \emptyset$ within known techniques to get directly convergence results as (1.6); controlled inexactness is an important requirement for existing inexact bundle methods.

The goal of this paper is to make the natural interest of accelerating bundle methods by using as much information as possible meet with convergence guarantees.

1.4. A generic scheme – and two instances – using uncontrolled bundle information. We sketch a basic generic bundle-type scheme using both fine and coarse information. It corresponds to what would do a practitioner using an algorithm and willing to incorporate more information available or cheap to compute. Many algorithms could be formulated within this abstract scheme; we will precisely study a simple one (Algorithm 1 in Section 2) and an advanced one (Algorithm 2 in Section 3 allowing a limited memory and converging without any assumptions).

After a Step 0 to initialize the parameters and variables, the general scheme consists in a loop between 3 steps:

- *Step 1: Construct new cutting-plane model calling the coarse oracle only.* In standard bundle-type methods, the cutting-plane model (1.10) is updated with adding a linearization (1.9) obtained by calling the η-oracle. In our setting, we consider that the cutting-plane model is constructed by an "external module" having the previous model as an input, and using only the coarse inexact oracle. For example, this external module can be the run of an inexact bundle method calling the coarse oracle. In Algorithm 1, Step 1 consists in running an inexact Kelley-method. In Algorithm 2, we consider the abstract case with a general external module producing a cutting-plane model.
- *Step 2: Test termination.* Assuming that the most expensive operation is to call the fine η-orcale, we test η-optimality before Step 3. Any standard test using the bundle information given by Step 1 and guaranteeing to have an approximate η-solution is appropriate for this step. In the two algorithms of this paper, the stopping test is based on testing approximate negativity of an inexact gap

$$\Delta_k = f_k^{\text{up}} - f_k^{\text{low}}, \tag{1.12}$$

  with a lower bound f_k^{low} and an inexact upper bound f_k^{up} such that

$$f_k^{\text{low}} \leq f_* \leq f_k^{\text{up}} + \eta \qquad \text{for all } k. \tag{1.13}$$

- *Step 3: Call fine oracle and compute next iterate.* This step consists in an iteration of a bundle-type algorithm: call of the η-oracle and definition of the next iterate. In Algorithm 1, Step 3 is an iteration of the type of the ones of the Kelley method. In Algorithm 2, Step 3 consists in a descent proximal iteration, similar to the one of [BKL95]. At iteration k of these algorithms, we have both a current iterate x_k and a current "best point" $\hat{x}_k$ (called "stability center" in the usual bundle terminology).

The difference between standard bundle-type algorithms and the ones considered in this paper is essentially contained in Step 1. The call of an external module makes us lose control on the construction of the cutting-plane model and therefore on the next iterate (see e.g. [HUL93]). The standard convergence proofs do not apply for the instances of this generic sheme.

1.5. Structure of the paper. The paper is structured as follows. Section 2 studies the simple Kelley-type algorithm following the above scheme, and proves (under a compactness assumption) its convergence up to η – which means that the sequence $\{\hat{x}_k\}$ generated by

the algorithm satisfies (1.6). Section 3 presents a general limited-memory proximal-descent level bundle algorithm following the scheme, and proves its convergence up to η (without any assumption). In Section 4, computational illustrations are exposed and discussed, to show that these methods save computational time in using both fine and coarse information.

2. Basic instance : Kelley method using two oracles. In this section, we study a simple instance of the general scheme presented in the introduction. The main ingredients of this method are an inexact Kelley method using the coarse oracle in Step 1, the test termination using the inexact gap Δ_k (1.12) in Step 2, and a simple descent-test in Step 3. Altogether, this gives the inexact Kelley-type method of Algorithm 1.

Algorithm 1 Inexact Kelley method using two oracles

▷ Step 0: Initialization

1: Choose $x_1 \in X$ and the stopping tolerance $\mathrm{tol}_\Delta > 0$
2: $(f_{x_1}, g_{x_1}) \leftarrow \eta$-oracle$(x_1)$,
3: Set $\hat{x}_1 gets x_1, f_1^{\mathrm{up}} \leftarrow f_{\hat{x}_1}, J_1^{\mathrm{f}} \leftarrow \{1\}, J_0^{\mathrm{c}} \leftarrow \emptyset$

4: **for** $k = 1, 2, \ldots$ **do**

▷ Step 1: Construct new cutting-plane model with coarse oracle

5: $(y^*, f_k^{\mathrm{low}}, J_k^{\mathrm{c}}) \leftarrow \mathrm{Kelley\_method}(J_k^{\mathrm{f}}, J_{k-1}^{\mathrm{c}})$
6: $\Delta_k \leftarrow f_k^{\mathrm{up}} - f_k^{\mathrm{low}}$

▷ Step 2: Test termination

7: **if** $\Delta_k \leq \mathrm{tol}_\Delta$ **then**
8: **return** $\hat{x}_k$ and $f_{\hat{x}_k} = f_k^{\mathrm{up}}$
9: **end if**

▷ Step 3: Call the η-oracle and compute next iterate

10: $x_{k+1} \leftarrow y^*$
11: $(f_{x_{k+1}}, g_{x_{k+1}}) \leftarrow \eta$-oracle$(x_{k+1})$
12: $J_{k+1}^{\mathrm{f}} \leftarrow J_k^{\mathrm{f}} \cup \{k+1\}$
13: **if** $f_{x_{k+1}} < f_k^{\mathrm{up}}$ **then**
14: $f_{k+1}^{\mathrm{up}} \leftarrow f_{x_{k+1}}$ and $\hat{x}_{k+1} \leftarrow x_{k+1}$
15: **else**
16: $f_{k+1}^{\mathrm{up}} \leftarrow f_k^{\mathrm{up}}$ and $\hat{x}_{k+1} \leftarrow \hat{x}_k$
17: **end if**

18: **end for**

When the run of the Kelley method of line 5 is stopped at the first iteration (i.e. only one minimization of the cutting-plane model), Algorithm 1 corresponds to an inexact Kelley method using only the η-oracle. Up to our knowledge, this basic method has not been studied so far in the litterature of inexact bundle methods.

The proof of the convergence of Algorithm 1 (encompassing the inexact Kelley method) is inspired from the one of the exact Kelley method of [HUL93, XII.4.2]. We also use the following technical lemma, valid in a more general context and used later in section 3.3.

LEMMA 2.1 (Nonpositivity of Δ_k and convergence). *Consider two sequences $\{f_k^{\mathrm{up}}\}$ and $\{f_k^{\mathrm{low}}\}$ satisfying* (1.13)*, and a sequence $\{\hat{x}_k\}$ such that $f_k^{\mathrm{up}} = f_{\hat{x}_k}$. Suppose that $\{f_k^{\mathrm{up}}\}$ is nonincreasing and $\{f_k^{\mathrm{low}}\}$ is nondecreasing. If $\lim \Delta_k \leq 0$, then the sequence $\hat{x}_k$ satisfies*

$$f_* - \eta \leq \lim f_{\hat{x}_k} \leq f_* \leq \liminf f(\hat{x}_k) \leq f_* + \eta\,. \tag{2.1}$$

Furthermore, if at some iteration k we have $\Delta_k \leq 0$, then we have in fact

$$f_* - \eta \leq f_{\hat{x}_k} \leq f_* \leq f(\hat{x}_k) \leq f_* + \eta\,. \tag{2.2}$$

Proof. Note first that the η-oracle properties imply that, for all k,

$$f_* - \eta \leq f(\hat{x}_k) - \eta \leq f_{\hat{x}_k}\,, \tag{2.3}$$

so that $f_k^{\text{up}} = f_{\hat{x}_k}$ satisfies (1.13). The nonincreasing sequence $\{f_{\hat{x}_k}\}$ is bounded from below thus converges and $\lim f_{\hat{x}_k} \geq f_* - \eta$. Similarly the nondecreasing $\{f_k^{\text{low}}\}$ is bounded from above by f_*, thus it also converges and $\lim f_k^{\text{low}} \leq f_*$. Writing $\lim \Delta_k \leq 0$ as

$$\lim f_{\hat{x}_k} - \lim f_k^{\text{low}} \leq 0$$

we obtain

$$f_* - \eta \leq \lim f_{\hat{x}_k} \leq f_*. \tag{2.4}$$

Now passing to the limit-inf in (2.3) and adding η, we also have

$$f_* \leq \liminf f(\hat{x}_k) \leq \lim f_{\hat{x}_k} + \eta \leq f_* + \eta.$$

Combining this inequalities with (2.4) gives the announced inequalities (2.1).

The argument leading to the second announced inequality (2.2) is exactly the same as above. For a fixed k, (2.3) and $\Delta_k \leq 0$ give $f_* - \eta \leq f_{\hat{x}_k} \leq f_*$, and adding η to (2.3) yields

$$f_* \leq f(\hat{x}_k) \leq f_{\hat{x}_k} + \eta \leq f_* + \eta\,.$$

Combining the inequalities gives (2.2). □

We are going to apply this lemma with the two bounds f_k^{up} and f_k^{low} of Algorithm 1; let us explicit their expression. The test of line 13 yields that the inexact upper bound f_k^{up} is the minimum of the f_{x_i} given by the η-oracle: for all k

$$f_k^{\text{up}} = f_{\hat{x}_k} = \min_{i \leq k} f_{x_i}. \tag{2.5}$$

After calling the Kelley method of line 5, the lower bound f_k^{low} is equal to the minimum of the cutting-plane model: by line 10, we have for all k

$$f_k^{\text{low}} = \check{f}_k(x_{k+1}) = \min_{x \in X} \check{f}_k(x) \quad \big(\leq \min_{x \in X} f(x) = f_* \big) \tag{2.6}$$

We are now in position to prove convergence of Algorithm 1.

THEOREM 2.2 (Convergence of the inexact Kelley method). *Set* tol_Δ *to zero in Algorithm 1. If X is a compact set, then the sequence testing optimality Δ_k becomes nonpositive (*$\lim \Delta_k \leq 0$*), and the iterates $\{\hat{x}_k\}$ generate an η-minimizing sequence; more precisely*

$$f_* - \eta \leq \lim f_{\hat{x}_k} \leq f_* \leq \liminf f(\hat{x}_k) \leq f_* + \eta\,.$$

Proof. By line 12, the models $\check{f}_k$ are defined as maximums over an increasing set of linearisations. So we have $\check{f}_{k+1}(x) \geq \check{f}_k(x)$ for all $x \in X$, and then f_k^{low} defined as (2.6) is a nondecreasing sequence bounded from above by f_*. By construction, f_k^{up} defined as (2.5) is

nonincreasing and bounded from below by $f_* - \eta$. Thus the assumptions of lemma 2.1 are enforced: we just have to show that $\lim \Delta_k \leq 0$ and the proof will be finished.

By monotonicity of f_k^{up} and f_k^{low}, we have that Δ_k is nonincreasing. For sake of a contradiction, assume that the sequence is bounded below by $\varepsilon > 0$. Take two indices $j < k$; then develop the fact that $\varepsilon \leq \Delta_{k-1}$ as follows:

$$\begin{array}{rcll} \varepsilon & \leq & f_{k-1}^{\text{up}} - f_{k-1}^{\text{low}} & \\ & = & f_{k-1}^{\text{up}} - \check{f}_{k-1}(x_k) & \text{by (2.6)} \\ & \leq & f_{x_j} - \check{f}_{k-1}(x_k) & \text{by (2.5)} \\ & \leq & f_{x_j} - (f_{x_j} + g_{x_j}^\top (x_k - x_j)) & \text{since } j \leq k-1 \\ & \leq & \|g_{x_j}\| \|x_k - x_j\| & \\ & \leq & \Lambda \|x_k - x_j\| & \end{array}$$

the last inequality coming from the fact that η-subgradients are bounded (by Λ) on X compact ([HUL93, Prop. XI.4.1.2]). Therefore we have

$$\|x_k - x_j\| \geq \frac{\varepsilon}{\Lambda} \qquad \text{for all } j < k.$$

This contradicts the fact that there exists a converging subsequence of the bounded sequence $\{x_k\} \subset X$. Thus we have $\lim \Delta_k \leq 0$. □

The inexact Kelley method of Algorithm 1 using two oracles (1.8) is thus η-convergent when X is compact. Using more information than standard Kelley method, we expect it to be faster in practice: this is indeed what we observe on the computational experiments of section 4.

3. Advanced instance : level method using two oracles. The algorithm of the previous section illustrates, in a very simple way, our scheme to use the coarse information. But it has two important drawbacks – a theoretical one and a practical one. First, its convergence is guaranteed only under a compactness assumption. Second, Kelley methods in general are known to be instable; bundle methods (proximal or level) are meant to stabilize them, and to have better practical efficiency.

We present in this section a bundle method following the generic sheme and answering to both of the above points: general convergence and practical performance. The algorithm is presented in Section 3.1, its convergence is stated in Section 3.2 and analyzed in Section 3.3. Its numerical behaviour is illustrated in Section 4.

3.1. An inexact proximal-descent level bundle method. This section presents Algorithm 2, following the scheme involving the two oracles in (1.8). The main novelty of this bundle method is the managment of coarse information ($J_k^c \neq \emptyset$).

Note that, disregarding coarse information (taking $J_k^c = \emptyset$), the algorithm corresponds to an inexact proximal-descent level bundle method, extending the one of [BKL95] to handle inexact oracles. Up to our knowledge, this is the first level-bundle method for unbounded feasible set with fixed inexact oracle; previous inexact level methods used indeed: a vanishing inexactness (see [OS13] and [Fáb00]) or a compact feasible set (as in the PhD dissertation of the second author).

Let us review the main ingredients of the algorithm.

– *External module.* Contrary to Algorithm 1, Step 1 is not specified here: it can be anything creating a cutting-plane model. The algorithm does not care either about what is done inside of the external module, nor about the returned candidate y^*. In the numerical experiments, we use a standard level method [LNN95] using the coarse oracle.

– *Level iteration.* We consider a level version of bundle methods, which give us a better control on the iterations (to compensate the lost of control on the construction of the model in Step 1). As in other level methods, the next iterate x_{k+1} is the projection of the current stability center $\hat{x}_k$ onto the polyedral "level set"

$$\mathbb{X}_k := \left\{ x \in X : \check{f}_k(x) \le f_k^{\text{lev}} \right\} = \left\{ x \in X : \bar{f}_i(x) \le f_k^{\text{lev}} \ \text{ for all } i \in J_k^{\text{f}} \cup J_k^{\text{c}} \right\} \tag{3.1}$$

with a level parameter $f_k^{\text{lev}} = f_k^{\text{up}} - v_k$ and a depth $v_k \ge 0$; in other words,

$$x_{k+1} = P_{\mathbb{X}_k}(\hat{x}_k) := \arg\min_{x \in \mathbb{X}_k} \frac{1}{2}\|x - \hat{x}_k\|^2. \tag{3.2}$$

Optimality conditions of this projection problem can be written, with the help of the Lagrange multipliers $\alpha_i \ge 0$ associated to the constraints $\bar{f}_i(x) \le f_k^{\text{lev}}$, as

$$-x + \hat{x}_k - \sum_{i \in J_k^{\text{f}} \cup J_k^{\text{c}}} \alpha_i g_i \ \in \ N_X(x) \quad \text{(the normal cone to } X \text{ at } x\text{)}.$$

Introducing the "stepsize" $\mu_k := \sum \alpha_i$, observe that x_{k+1}, the unique solution of the above optimality conditions, can be written as the "subgradient step"

$$x_{k+1} = \hat{x}_k - \mu_k \hat{g}_k \qquad (\text{such that} \quad \mu_k(\check{f}_k(x_{k+1}) - f_k^{\text{lev}}) = 0\,), \tag{3.3}$$

along the direction $\hat{g}_k \in \partial \check{f}_k(x_{k+1}) + N_X(x_{k+1})$ called "aggregated subgradient" (which will have a role in the stopping tests). In practice, the problem (3.2) is solved at line 16 by a quadratic programming solver which provides x_{k+1} and μ_k, from which we deduce $\hat{g}_k$ by (3.3). If $\mathbb{X}_k$ is empty, there holds $f_k^{\text{lev}} < \check{f}_k(x) \le f(x)$ for all $x \in X$, and we exploit this information: when the quadratic programming solver raises a flag of empty set, we can update $f_k^{\text{low}} = f_k^{\text{lev}}$ the lower bound for the optimal value f_* (see line 14).

– *Descent step.* To avoid any compactness assumption in our convergence analysis, we consider a proximal-descent version of level bundle method, inspired from the one of [BKL95]. The stability center is updated only if the observed decrease is at least a fraction of the level depth, that is,

$$f_{x_{k+1}} \le f_{\hat{x}_k} - \kappa_f v_k\,, \quad \text{with} \quad \kappa_f \in (0,1)\,. \tag{3.4}$$

– *Inexactness.* Proximal bundle methods face excessive inexactness by increasing the stepsizes (see [Kiw06] or the recent [OSL13]). In level methods, the stepsize is a Lagrange multiplier on which we have not a direct control. Thus, the level bundle algorithm of [BKL95] does not handle inexact oracles. In contrast, our algorithm does handle a fixed inexact oracle, together with the uncontrolled coarse inexactness. More precisely, inexactness of the oracle is handled at line 22: roughly speaking, we do not decrease the depth v_k when the noise is excessively large compared to g_k. Note also that our convergence proof differs from the one of [BKL95]. As a result, we obtain convergence without any compactness-like assumption.

– *Bundle compression.* We use cutting-plane models $\check{f}_k$ incorporating both fine and coarse linearizations, which are possibly numerous and not so precise. It is then important to be able to work with a limited memory and to somehow extract the useful part from all the bundle information. In bundle algorithm terminology, this is called "bundle compression", which is a desirable property in general [HUL93], and thus even more in our context. We

Algorithm 2 Inexact proximal level method using two oracles

▷ Step 0: initialization

1: Choose $x_1 \in X$, set $\hat{x}_1 \leftarrow x_1$
2: Choose stopping tolerances $\mathrm{tol}_\Delta > 0$, $\mathrm{tol}_e > 0$ and $\mathrm{tol}_g > 0$
3: Select $\kappa_l\, \kappa_f, \kappa_\delta, \kappa_{\mathrm{att}} \in (0,1)$
4: Choose an upper bound $\mu_{\max} > 0$
5: $(f_{x_1}, g_{x_1}) \leftarrow \eta$-oracle$(x_1)$, set $\hat{g}_1 \leftarrow g_{x_1}$ and $\hat{e}_1 \leftarrow 0$
6: Set $f_1^{\mathrm{up}} \leftarrow f_{x_1}$, $f_1^{\mathrm{low}} \leftarrow -\infty$, $\Delta_1 \leftarrow +\infty$, $J_1^{\mathrm{f}} \leftarrow \{1\}$, $J_0^{\mathrm{c}} \leftarrow \emptyset$
7: **for** $k = 1, 2, \ldots$ **do**

▷ Step 1: Construct new cutting-plane model with coarse oracle

8: $(y^*, f_{y^*}, J_k^{\mathrm{c}}) \leftarrow$external_module$(\hat{x}_k, J_k^{\mathrm{f}}, J_{k-1}^{\mathrm{c}})$ ▷ abstract external module
9: **if** $k = 1$ **then**
10: $v_1 \leftarrow f_1^{\mathrm{up}} - f_{y^*}$
11: **end if**

▷ Step 2: Test termination

12: Update $f_k^{\mathrm{lev}} \leftarrow f_k^{\mathrm{up}} - v_k$ and $\mathbb{X}_k \leftarrow \left\{x \in X : \check{f}_k(x) \le f_k^{\mathrm{lev}}\right\}$ ▷ level management
13: **if** $\mathbb{X}_k$ is empty **then**
14: $f_k^{\mathrm{low}} \leftarrow f_k^{\mathrm{lev}}$, $\Delta_k \leftarrow f_k^{\mathrm{up}} - f_k^{\mathrm{low}}$, $v_k \leftarrow (1-\kappa_l)\Delta_k$ ▷ lower bound
15: **else**
16: solve (3.2) to get x_{k+1} and μ_k, and compute $\hat{g}_k$ ▷ projection
17: $\hat{e}_k \leftarrow v_k - \mu_k \|\hat{g}_k\|^2$
18: **end if**
19: **if** $\Delta_k \le \mathrm{tol}_\Delta$ or ($\hat{e}_k \le \mathrm{tol}_e$ and $\|\hat{g}_k\| \le \mathrm{tol}_g$) **then** ▷ stopping test
20: **return** $\hat{x}_k$ and $f_{\hat{x}_k} = f_k^{\mathrm{up}}$
21: **end if**
22: **if** $\mu_k > \mu_{\max}$ and $\hat{e}_k \ge -\kappa_{\mathrm{att}} \mu_k \|\hat{g}_k\|^2$ **then** ▷ attenuation
23: $v_k \leftarrow \frac{v_k}{2}$, and go back to line 12
24: **end if**

▷ Step 3: Call the η-oracle, and compute next iterate

25: $(f_{x_{k+1}}, g_{x_{k+1}}) \leftarrow \eta$-oracle$(x_{k+1})$ ▷ call fine oracle
26: **if** $f_{x_{k+1}} \le f_{\hat{x}_k} - \kappa_f v_k$ **then**
27: $\hat{x}_{k+1} \leftarrow x_{k+1}$, $f_{k+1}^{\mathrm{up}} \leftarrow f_{\hat{x}_{k+1}}$ and $f_{k+1}^{\mathrm{low}} \leftarrow f_k^{\mathrm{low}}$ ▷ descent step
28: $\Delta_k \leftarrow f_{k+1}^{\mathrm{up}} - f_{k+1}^{\mathrm{low}}$ and $v_{k+1} \leftarrow \min\{v_k, (1-\kappa_l)\Delta_k\}$
29: **else**
30: $\hat{x}_{k+1} \leftarrow \hat{x}_k$, $\Delta_{k+1} \leftarrow \Delta_k$, $v_{k+1} \leftarrow v_k$, $f_{k+1}^{\mathrm{up}} \leftarrow f_k^{\mathrm{up}}$ and $f_{k+1}^{\mathrm{low}} \leftarrow f_k^{\mathrm{low}}$ ▷ null step
31: **end if**
32: Choose $J_{k+1}^{\mathrm{f}} \supset \{k+1, -k\}$ and update J_k^{c} ▷ bundle compression
33: **end for**

emphasize that in theory we can compress a lot: are enough (see line 32) only the current fine information and the so-called "aggregate linearization"

$$\bar{f}_{-k}(\cdot) := \check{f}_k(x_{k+1}) + \hat{g}_k^\top(\cdot - x_{k+1}) \tag{3.5}$$

which can be proved (see e.g. [OS13, Prop. 3.2]) to be a linearization itself

$$\bar{f}_{-k}(x) \le \check{f}_k(x) \le f(x) \qquad \text{for all } x \in X\,. \tag{3.6}$$

– *More aggregated objects.* We use the convenient notation "$-k$" for objects related to the aggregate linearization at iteration k (see e.g. [OSL13]). We define the "aggregate level set"

$\mathbb{X}_{-k} := \{x \in X : \bar{f}_{-k}(x) \leq f_k^{\text{lev}}\}$, which can be proved (see [OS13, Prop. 3.2]) to produce the same iterate that the (complete) level set $\mathbb{X}_k$; in other words,

$$x_{k+1} = P_{\mathbb{X}_k}(\hat{x}_k) = P_{\mathbb{X}_{-k}}(\hat{x}_k) \quad \text{and} \quad (\hat{x}_k - x_{k+1})^\top (x - x_{k+1}) \leq 0 \quad \text{for all} \quad x \in \mathbb{X}_{-k}\,. \tag{3.7}$$

We also define the "aggregated linearization error" by

$$\hat{e}_k := f_{\hat{x}_k} - \bar{f}_{-k}(\hat{x}_k) \quad \big(\geq -\eta\big); \tag{3.8}$$

the inequality coming from (1.3) and (3.6) since: $\hat{e}_k \geq f(\hat{x}_k) - \eta - \bar{f}_{-k}(\hat{x}_k) \geq -\eta$. In addition to the (inexact) optimality gap Δ_k, an additional certificate of optimality makes use of $\hat{e}_k$ (see line 19 and lemma 3.2 below). Finally we note that the aggregated linearization error can be computed from the agregated subgradient and the level depth: more precisely, when $\mu_k > 0$, we have

$$\hat{e}_k = v_k - \mu_k \|\hat{g}_k\|^2\,. \tag{3.9}$$

To see this, notice from (3.3) that $\mu_k > 0$ ensures that $\check{f}_k(x_{k+1}) = f_k^{\text{lev}}$ and, therefore

$$\hat{e}_k = f_{\hat{x}_k} - (\check{f}_k(x_{k+1}) + \hat{g}_k^\top(\hat{x}_k - x_{k+1})) = f_{\hat{x}_k} - f_k^{\text{lev}} - \mu_k\|\hat{g}_k\|^2 = v_k - \mu_k\|\hat{g}_k\|^2\,.$$

3.2. Convergence result. We have the following theorem stating the convergence of Algorithm 2, which is of the same vein as Theorem 2.2 for Algorithm 1.

THEOREM 3.1 (Convergence of inexact proximal level). *Set the tolerances at zero in Algorithm 2. Then the sequences testing optimality $\{\Delta_k\}$, $\{\hat{e}_k\}$ and $\{\hat{g}_k\}$ become "nonpositive", in the sense that*

- *either the sequence $\{\Delta_k\}$ tends to be nonpositive:* $\lim \Delta_k \leq 0$,
- *or there exists a subsequence (indexed by $\mathscr{I}$) such that:* $\liminf_{k\in\mathscr{I}} \hat{e}_k \leq 0$ *and* $\lim_{k\in\mathscr{I}} \|\hat{g}_k\| = 0$.

Furthermore, the iterates $\{\hat{x}_k\}$ generate a η-minimizing sequence, i.e.

$$f_* \leq \liminf f(\hat{x}_k) \leq f_* + \eta \tag{3.10}$$

Thus Algorithm 2 terminates after finitely many steps with an approximate solution if the tolerances tol_Δ, tol_g, *and* tol_e *are strictly positive.*

The next section is devoted to the proof of this theorem. We will say that the algorithm converges up to η when (3.10) holds.

Before moving on to the proof of η-convergence, let us take a closer look at the stopping tests of Algorithm 2. We see that $\{f_k^{\text{up}} = f_{\hat{x}_k}\}$ is nonincreasing (line 27), $\{f_k^{\text{low}}\}$ is nondecreasing (line 14), and $\{\Delta_k\}$ is nonincreasing (line 28). By Lemma 2.1, we have that

$$\lim \Delta_k \leq 0 \quad \Longrightarrow \text{ convergence up to } \eta. \tag{3.11}$$

This leads us to the first stopping test of line 19, the same as the one of Algorithm 1. Without some compactness assumption on the problem, we cannot guarantee this stopping test to hold. Then we need a second stopping test (see line 19) based on the agregated error and subgradients; the next lemma explains its consistency.

LEMMA 3.2 (Vanishing aggregated errors and convergence). *For the sequences $\{\hat{x}_k\}$, $\{\hat{e}_k\}$ and $\{\hat{g}_k\}$ generated by Algorithm 2, we have, for all $x \in X$,*

$$f(\hat{x}_k) \leq f(x) + \hat{e}_k + \eta - \hat{g}_k^\top(x - \hat{x}_k). \tag{3.12}$$

Assume that $\{\hat{x}_k\}$ is bounded and that there exists a subsequence indexed by $\mathscr{I} \subset \{1,2,\ldots\}$ such that

$$\liminf_{k\in\mathscr{I}} \hat{e}_k \leq 0 \quad \textit{and} \quad \lim_{k\in\mathscr{I}} \|\hat{g}_k\| = 0. \tag{3.13}$$

Then the algorithm is convergent up to η.

Proof. Fix $x \in X$. The inequality (3.12) comes from (3.5) as follows:

$$\begin{aligned} f(x) &\geq \bar{f}_{-k}(x) \\ &= \check{f}_k(x_{k+1}) + \hat{g}_k^\top(x - x_{k+1}) \\ &= \check{f}_k(x_{k+1}) + \hat{g}_k^\top(\hat{x}_k - x_{k+1}) + \hat{g}_k^\top(x - \hat{x}_k) \\ &= \bar{f}_{-k}(\hat{x}_k) + \hat{g}_k^\top(x - \hat{x}_k) \\ &= f_{\hat{x}_k} - (f_{\hat{x}_k} - \bar{f}_{-k}(\hat{x}_k)) + \hat{g}_k^\top(x - \hat{x}_k) \\ &= f_{\hat{x}_k} - \hat{e}_k + \hat{g}_k^\top(x - \hat{x}_k) \\ &\geq f(\hat{x}_k) - \eta - \hat{e}_k + \hat{g}_k^\top(x - \hat{x}_k). \end{aligned}$$

We also get the upper bound

$$f_* \leq f(\hat{x}_k) \leq f(x) + \hat{e}_k + \eta + \|\hat{g}_k\|\|x - \hat{x}_k\|.$$

Passing to the lim inf, the assumption (3.13) together with the boundedness of $\{\hat{x}_k\}$ yields

$$f_* \leq \liminf_{k\in\mathscr{I}} f(\hat{x}_k) \leq f(x) + \eta.$$

Taking the infimum over $x \in X$ gives (3.10). □

3.3. Convergence proof. To prove Theorem 3.1, we adapt the usual rationale of convergence proof of bundle methods, by considering the two cases of infinitely many and finitely many descent steps (line 27). We show that in both cases one of the two stopping tests is active, which guarantees in turn that the algorithm converges up η (by (3.11) and Lemma 3.2). The technical challenge is to handle, first, a fixed inexactness in a level method and, second, the uncontrolled cutting-plane model.

We start with a remark about the level depth v_k in the algorithm. Looking at lines 14, 23 and 28, we see that $\{v_k\}$ is nonincreasing, and that if $v_k \geq 0$ then $\hat{e}_k \geq -\mu_k\|\hat{g}_k\|^2$. We also notice that if $v_k < 0$ then we have $\Delta_k < 0$ and then (3.11). Therefore, we consider that $v_k \geq 0$ in the remainder of the section.

We will also need the following two index sets:

- the set $\mathscr{A}$ of the iterations requiring a level attenuation (line 23),
- the set $\mathscr{K}$ of the iterations for which $v_k = (1-\kappa_l)\Delta_k$.

The next lemma studies the situation of infinitely many level attenuations, and the following proposition treats the first case of infinitely many descent steps.

LEMMA 3.3 (Infinitely many level attenuations). *If $\mathscr{A}$ contains infinitely many indices, then* (3.13) *holds with $\mathscr{I} = \mathscr{A}$. If the sequence $\{\hat{x}_k\}$ is furthermore bounded, then the algorithm converges up to η.*

Proof. Recall that $v_k = \hat{e}_k + \mu_k\|\hat{g}_k\|^2$ by (3.9). If $k \in \mathscr{A}$, then we have

$$v_k = \hat{e}_k + \mu_k\|\hat{g}_k\|^2 \geq (1-\kappa_{\text{att}})\mu_k\|\hat{g}_k\|^2 \geq (1-\kappa_{\text{att}})\mu_{\max}\|\hat{g}_k\|^2 \geq 0.$$

If the set $\mathscr{A}$ is infinite, then we have $v_k \to 0$, and therefore $\|\hat{g}_k\| \to 0$ by the above inequality. By (3.9), this yields that $\hat{e}_k \to 0$ and then we have (3.13) with $\mathscr{I} = \mathscr{A}$. As a result, if the sequence $\{\hat{x}_k\}$ is bounded, we can invoke Lemma 3.2 and get that $\{\hat{x}_k\}$ is η-minimizing. □

PROPOSITION 3.4 (Infinitely many descent steps). *Suppose there are infinitely many descent steps (line 27). Then the algorithm converges up to η.*

Proof. Let us index the descent steps by ℓ. More precisely $k(\ell)$ denotes the ℓ^{th} descent iteration, and $j(\ell) = k(\ell+1)-1$ the last iteration before the $(\ell+1)^{\text{th}}$. Note that $\hat{x}_{k(\ell)}$ is the ℓ^{th} (different) stability center, and that $\hat{x}_{k(\ell)} = \hat{x}_{j(\ell)}$. The descent step (3.4) gives the inequality

$$f_{x_{k(\ell)}} - f_{x_{k(\ell+1)}} \geq \kappa_f v_{j(\ell)} \geq 0\,.$$

Summing over ℓ we get

$$f_{x_{k(0)}} - \lim_{\ell} f_{x_{k(\ell+1)}} \geq \kappa_f \sum_{\ell=0}^{\infty} v_{j(\ell)}\,.$$

Since $\lim_\ell f_{x_{k(\ell+1)}} \geq f_* - \eta > -\infty$, we get that the serie converges and then

$$\lim_{\ell} v_{j(\ell)} = 0. \tag{3.14}$$

By monotonicity of v_k, we thus have $\lim_k v_k = 0$, and therefore either

$$\lim_{k\in\mathscr{K}} v_k = 0 \quad or \quad \lim_{k\in\mathscr{A}} v_k = 0.$$

In the first case, we have $\lim_{k\in\mathscr{K}} \Delta_k = 0$, by definition of $\mathscr{K}$. Thus, (3.11) holds and the proof is over.

In the second case, Lemma 3.3 ensures first (3.13). If $\{\hat{x}_k\}$ is bounded, it also ensures that $\{\hat{x}_k\}$ is η-minimizing. Let us focus on the case when $\{\hat{x}_k\}$ is unbounded, and let us prove by contradiction that $\{\hat{x}_k\}$ is still η-minimizing.

Suppose that there exists $\varepsilon > 0$ such that $f(\hat{x}_k) > f_* + \eta + \varepsilon$ for all k large enough. This yields that there exists $\tilde{x} \in X$ such that $f(\hat{x}_{k(\ell)}) \geq f(\tilde{x}) + \eta + \varepsilon/2$ for all large ℓ. Then (3.12) applied to $k = j(\ell)$ gives

$$\hat{g}_{j(\ell)}^\top (x - \hat{x}_{k(\ell)}) \leq f(x) + \eta - f(\hat{x}_{k(\ell)}) + \hat{e}_{j(\ell)} \quad \text{for all} x \in X\,,$$

which yields

$$\hat{g}_{j(\ell)}^\top (\tilde{x} - \hat{x}_{k(\ell)}) \leq \hat{e}_{j(\ell)} - \varepsilon/2\,.$$

Using this inequality and (3.9), we develop

$$\begin{aligned}
\|\hat{x}_{k(\ell+1)} - \tilde{x}\|^2 &= \|\hat{x}_{k(\ell)} - \mu_{j(\ell)}\hat{g}_{j(\ell)} - \tilde{x}\|^2 \\
&= \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + \|\mu_{j(\ell)}\hat{g}_{j(\ell)}\|^2 + 2\mu_{j(\ell)}\hat{g}_{j(\ell)}^\top(\tilde{x} - \hat{x}_{k(\ell)}) \\
&= \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + \mu_{j(\ell)}[\mu_{j(\ell)}\|\hat{g}_{j(\ell)}\|^2 + 2\hat{g}_{j(\ell)}^\top(\tilde{x} - \hat{x}_{k(\ell)})] \\
&\leq \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + \mu_{j(\ell)}[\mu_{j(\ell)}\|\hat{g}_{j(\ell)}\|^2 + 2\hat{e}_{j(\ell)} - \varepsilon] \\
&\leq \|\hat{x}_{k(\ell)} - \tilde{x}\|^2 + 2\mu_{j(\ell)}[v_{j(\ell)} - \varepsilon/2]\,.
\end{aligned}$$

As $\lim_\ell v_{j(\ell)} = 0$ by (3.14), we have for all ℓ large enough $v_{j(\ell)} \leq \varepsilon/2$ and then

$$\|\hat{x}_{k(\ell+1)} - \tilde{x}\|^2 \leq \|\hat{x}_{k(\ell)} - \tilde{x}\|^2$$

which contradicts the fact that the sequence $\{\hat{x}_k\}$ is unbounded. Hence, (3.10) must hold. □

We consider now the second case of finitely many descent steps. We start with a lemma showing that null iterates get further away from the last stability center.

LEMMA 3.5 (After a last descent step). *If $\hat{x}_k = \hat{x}_{k-1} = \hat{x}$, $f_k^{\mathrm{lev}} \leq f_{k-1}^{\mathrm{lev}}$ and $v_k = v_{k-1}$, then we have*

$$\|x_{k+1} - \hat{x}\|^2 \geq \|x_k - \hat{x}\|^2 + \frac{(1-\kappa_f)^2}{\|g_{x_k}\|^2} v_k^2.$$

Proof. The bundle management strategy at line 32 incorporates two pieces: the k-th linearization ($\bar{f}_k$) and the aggregate linearisation ($\bar{f}_{-k}$) in the model $\check{f}_k$. We thus distinguish two parts in the proof.

Part 1: Since $\bar{f}_{-(k-1)} \leq \check{f}_k$ and $f_k^{\mathrm{lev}} \leq f_{k-1}^{\mathrm{lev}}$, we have $\mathbb{X}_k \subset \mathbb{X}_{-(k-1)}$, and therefore $x_{k+1} \in \mathbb{X}_{-(k-1)}$. Apply now (3.7) to get

$$(\hat{x} - x_k)^\top (x_{k+1} - x_k) \leq 0.$$

Developing $\|x_{k+1} - \hat{x}\|^2 = \|x_{k+1} - x_k + (x_k - \hat{x})\|^2$, the inequality gives

$$\|x_{k+1} - \hat{x}\|^2 \geq \|x_k - \hat{x}\|^2 + \|x_k - x_{k+1}\|^2$$

We just have to prove now that

$$\|x_{k+1} - x_k\| \geq \frac{(1-\kappa_f)}{\|g_{x_k}\|} v_k. \tag{3.15}$$

Part 2: Since $\bar{f}_k \leq \check{f}_k$ and $x_{k+1} \in \mathbb{X}_k$, we have $f_{x_k} + g_{x_k}^\top (x_{k+1} - x_k) \leq f_k^{\mathrm{lev}}$, which gives

$$f_{x_k} - f_k^{\mathrm{lev}} \leq \|g_{x_k}\| \|x_{k+1} - x_k\|. \tag{3.16}$$

Iteration k is not an descent iteration: the converse of line 26 reads $f_{x_k} \geq f_{\hat{x}} - \kappa_f v_{k-1}$. Recalling that $f_k^{\mathrm{lev}} = f_{\hat{x}} - v_k$ and $v_k = v_{k-1}$, this yields $f_{x_k} - f_k^{\mathrm{lev}} \geq (1-\kappa_f) v_k$. Together with (3.16), this gives (3.15), and ends the proof. □

PROPOSITION 3.6 (Finitely many descent steps). *Suppose that Algorithm 2 generates only finitely many descent steps. Then the algorithm converges up to η.*

Proof. Let us consider first two easy cases. If $\lim \Delta_k \leq 0$ then (3.11) holds, and the proof is over. If $\mathscr{A}$ has infinitely many indices, we can conclude with Lemma 3.3 together with the fact that the sequence $\{\hat{x}_k\}$ is constant for k large enough.

Let us focus on the case where there exists $\bar{\Delta} > 0$ such that $\Delta_k \geq \bar{\Delta}$ for all k, and there is eventually no noise attenuation (the set $\mathscr{A}$ has finitely many indices). For k large enough, the stability center is fixed (denoted $\hat{x}$) and the level depth is also fixed (at $\bar{v} > 0$).

We claim that the sequence $\|x_{k+1} - \hat{x}\|$ is not bounded. For sake of a contradiction, suppose that it is bounded. Then the η-subgradients are bounded (by a constant Λ) by [HUL93, Prop. XI.4.1.2]. Apply Lemma 3.5, since the v_k and the f_k^{lev} are fixed: the sequence $\|x_{k+1} - \hat{x}\|$ increase by a constant factor $(1-\kappa_f)^2 \bar{v}^2 / \Lambda^2$ at each iteration. This contradicts the boundedness.

We claim now that $\mu_k \to \infty$. In order to get a contradiction, suppose that $\{\mu_k\}$ is bounded: there exists $\bar{\mu} > 0$ such that $\mu_k \leq \bar{\mu}$ for all k large enough. Using (3.9) we have that

$$\mu_k v_k = \mu_k \hat{e}_k + \mu_k^2 \|\hat{g}_k\|^2 \geq -\mu_k \eta + \mu_k^2 \|\hat{g}_k\|^2 \geq -\bar{\mu}\eta + \|x_{k+1} - \hat{x}\|^2.$$

As $\{v_k\}$ is nonincreasing, we have that $\bar{\mu} v_0 \geq \mu_k v_k \geq -\bar{\mu}\eta + \|x_{k+1} - \hat{x}\|^2$, contradicting that $\|x_{k+1} - \hat{x}\|^2 \to \infty$. Hence, $\mu_k \to \infty$.

Since there is eventually no noise attenuation, we have (see line 22)

$$\hat{e}_k < -\kappa_{\mathrm{att}} \mu_k \|\hat{g}_k\|^2 < 0 \qquad \text{for all } k \text{ large enough.}$$

By (3.8), this yields $\|\hat{g}_k\|^2 \leq \eta / (\kappa_{\mathrm{att}} \mu_k)$. Since $\mu_k \to \infty$, we get that $\hat{g}_k \to 0$. Hence, (3.13) holds with $\mathscr{I}$ being all the large indices. Since the sequence $\{\hat{x}_k\}$ is finite (thus bounded), we can conclude with Lemma 3.2. □

4. Numerical illustration. In this section, we illustrate the efficacy of our scheme on a standard stochastic optimization problem. Our goal is not to obtain the best computational results for these problems, but to show that our scheme to use coarse bundle information does speed-up computations. Specifically, we compare Algorithm 1 and Algorithm 2 to their respective basic versions which do not use additional coarse information (namely, the Kelley method and the level bundle method, respectively).

4.1. Description of the numerical tests. We consider a set of two-stage stochastic linear test-problems (see e.g. [SDR09, Deá06, OSS11]). These problems are available on line at the webpage of István Deák, `http://web.uni-corvinus.hu/~ideak1/kut_en.htm`. Let us describe them briefly.

Each of these problems has the form (1.1) with

$$f(x) = c^\top x + \sum_{i=1}^{N} p_i f_i(x) \quad \text{and} \quad X = \{x \in \mathbb{R}^n_+ : Ax = b\},$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m_1 \times n}$ and $b \in \mathbb{R}^{m_1}$ are such that the set X is bounded. Also,

$$f_i(x) := \min_{y \in \mathbb{R}^{n_2}_+} q^\top y \ \text{ s.t. } \ Tx + Wy = h_i \tag{4.1}$$

is the so-called recourse function associated with the i-th scenario $h_i \in \mathbb{R}^{m_2}$ (which has a probability $p_i > 0$). In these problems, the vectors h_i are the only uncertainty parameters and are normally distributed. The dual linear problem of (4.1) is

$$f_i(x) = \sup_{W^\top u \le q} (h_i - Tx)^\top u. \tag{4.2}$$

A family of problems is given by the data (c, A, b, q, T, W) along with a generator of appropriate scenarios, which takes as an input the number of scenarios N, and returns (p_i, h_i) for $i = 1, \ldots, N$. We got on the webpage of István Deák 9 families of problems of different sizes; we call them F1 to F9. For each family, we have 7 problems corresponding to a number of scenarios $N \in \{100, 200, 500, 800, 1000, 1200, 1500\}$.

We use these tests-problems to illustrate our scheme using both fine and coarse bundle information. As explained in the introduction, computing exact information on f requires to solve the N linear optimization subproblems (4.1)-(4.2), but solving only a fraction of these subproblems still gives inexact information of f: the optimal solution $\bar{u}_i$ giving $f_i(x)$ can also be used to under-approximate other terms $f_j(x)$ (since the dual feasible sets are the same, we have $(d_j - T_i x)^\top \bar{u}_i \le f_j(x)$). Therefore, for a fixed fraction of solved problems, we have inexact information but with a unknown accuracy.

Thus we consider the following two oracles:

- a fine oracle providing the (exact) value $f(x)$ and a subgradient $g \in \partial f(x)$ ($\eta = 0$) by solving exactly the N subproblems (4.2)
- a coarse oracle computing an under approximation of f and an approximate subgradient, by solving $\frac{N}{5}$ of the subproblems (4.2) and taking a feasible solution of the remaining subproblems. This oracle is about five times faster than the fine one, but we do not control its accuracy.

We have implemented in MATLAB (using the ILOG CPLEX solver) the algorithms 1 and 2, along with their basic counterparts (taking $J_c^k = \emptyset$). In Algorithm 2, we used an inexact level method for the external module creating the uncontrolled cutting-plane model. In practice, we used a relative stopping tolerance of 10^{-3} for Algorithm 1 (and the Kelley method) and 10^{-5} for Algorithm 2 (and the level method). Note that since the fine oracle is exact ($\eta = 0$), all four methods converge to the exact solution, up to the stopping tolerance.

4.2. Numerical results. We test the four algorithms on the above 63 problems; we measure the CPU time and the number of calls to the fine oracle to reach convergence. Our goal is to compare the algorithms two by two (Algorithm 1 vs its basic version, and Algorithm 2 vs its basic version) to illustrate the interest of including coarse bundle information following our scheme.

Table 4.1 reports the (minimum, average and maximum) reduction on CPU time and fine oracle calls of Algorithm 1. First, we observe a substantial decrease in fine oracle calls for all the problems and a decrease in running time for the majority of the problems $(50/63)$. For a few instances we see an increase in the running time despite the decrease in fine oracle calls; this is due to the numerous calls to the coarse oracle, which for these instances, may have given very poor information. Note that on average, we observed more than ten times more calls to the coarse oracle than to the fine one. This also explains why the running time does not decrease in the same proportion as fine oracle calls.

Table 4.2 reports the reduction of CPU time and fine oracle calls for Algorithm 2. We observe a behaviour similar to the one showed by Table 4.1 with a global decrease in both CPU time and fine oracle calls. We see though that the reduction in oracle calls of Algorithm 2 is less important than the one of Algorithm 1, because the level method is already an efficient optimized algorithm. Finally, we noticed that the number of coarse oracle calls has the same order as the number of fine oracle calls.

	CPU time reduction (%)			Exact calls reduction (%)		
Problem family	min	avg	max	min	avg	max
F1	31	44	54	56	79	87
F2	12	35	48	62	73	75
F3	11	33	54	57	75	90
F4	26	47	60	52	81	90
F5	40	50	61	71	83	87
F6	9	32	47	51	77	87
F7	-18	12	48	47	71	96
F8	-18	-4	36	38	63	94
F9	-17	1	38	38	60	91
Total average	8	28	50	52	74	88

TABLE 4.1
Reduction of CPU time and exact oracle calls of Algorithm 1 compared to exact Kelley method

	CPU time reduction (%)			Exact calls reduction (%)		
Problem family	min	avg	max	min	avg	max
F1	23	31	51	42	53	72
F2	-5	15	38	16	42	66
F3	5	20	30	17	33	41
F4	10	27	45	33	42	50
F5	19	29	39	40	49	57
F6	8	15	28	20	25	35
F7	11	19	31	15	30	42
F8	0	14	33	8	24	44
F9	-6	5	17	10	18	24
Total average	7	19	35	22	35	48

TABLE 4.2
Reduction of CPU time and exact oracle calls of Algorithm 2 compared to exact level method

To compare speed and robustness of the algorithms globally on all the problems, we use the performance profiles introdced by [DM02]. For each algorithm, we plot the proportion of problems that it solved within a factor of the time required by the best algorithm. More precisely, if we denote by $t_A(p)$ the time spent by algorithm A to solve problem p and $t^*(p)$ the best time for solving problem p, then the proportion of problems solved by A within a factor τ is

$$\theta_A(\tau) = \frac{\text{number of problems } p \text{ such that } t_A(p) \leq \tau t^*(p)}{\text{total number of problems}}.$$

Figure 4.1 presents the performance profile of Algorithm 1 vs Kelley algorithm. Since its curve is always higher, Algorithm 1 clearly dominates. The values at $\tau = 1$ indicate that Algorithm 1 is the fastest to solve almost 80% of the problems. Then it solves 100% of the problems within a factor $\tau \approx 1.2$ of the best times, whereas Kelley algorithm needs $\tau \approx 2.6$ times the best times. Thus Algorithm 1 is much more robust than Kelley algorithm.

Figure 4.2 gives the performance profile of Algorithm 2 vs level algorithm. Similarly to the previous comparison, we see that Algorithm 2 dominates: it is faster for more than 90% of the problems, and solves all the problems within 1.4 times the best time, while level algorithm needs almost 2 times longer.

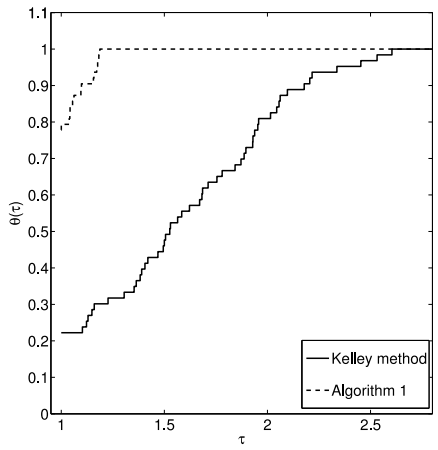

FIGURE 4.1. *Performance profiles for Kelley algorithms*

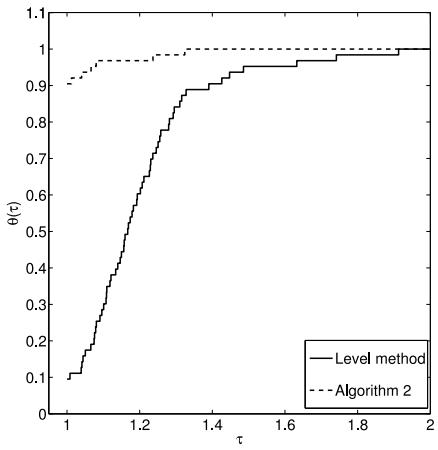

FIGURE 4.2. *Performance profiles for level algorithms*

5. Conclusions. This paper presents a scheme to extend inexact bundle-type algorithms to incorporate (already available or cheap to compute) uncontrolled bundle information. We formalize this additional information as a second inexact oracle, required only to give underlinearizations of the objective function, without bounded inexactness (which is, in contrast, an important requirement for other inexact bundle methods, see e.g. [Kiw06], [OS13], [OSL13] among others). We study two algorithms using this uncontrolled bundle information and instanciating the generic sheme; we also illustrate numerically that they are faster that their basic versions using only controlled bundle information.

This paper is mainly methodological and theoretical: beside the consideration of uncontrolled bundle information, the main contribution is the proof of convergence of the second algorithm (proximal-level bundle method using two oracles) which has several nice features,

including a limited memory and a convergence without any assumption. The technical challenge of this convergence proof was to manage the inexact oracles, the fixed inexactness of the first and the uncontrolled of the second. It is likely that other bundle-type algorithms could be proved to be convergent following the same rationale.

Beyond provably convergent algorithms, we finish with a practical comment. Recall that for Lagrangian relaxations of combinatorial optimization problems, and for decompositions of stochastic optimization problems, the call of the uncontrolled oracle is often neglectable compared to the call of the fine one. In this case, a wise practitioner can be tempted to use the uncontrolled bundle information readily inside of his current bundle method (implementing a basic version of the generic sheme). We hope that this paper can serve as an incentive to follow this meaningful practical intuition, as it establishes that incorporating uncontrolled bundle information is also coherent in theory.

REFERENCES

[BKL95] U. Brannlund, K. C. Kiwiel, and P. O. Lindberg, *A descent proximal level bundle method for convex nondifferentiable optimization*, Operations Research Letters **17** (1995), no. 3, 121 – 126.

[Deá06] I. Deák, *Two-stage stochastic problems with correlated normal variables: computational experiences*, Annals OR **142** (2006), no. 1, 79–97.

[DM02] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming **91** (2002), 201–213.

[Fáb00] C. Fábián, *Bundle-type methods for inexact data*, Central European Journal of Operations Research **8** (2000), 35–55.

[Fra02] A. Frangioni, *Generalized bundle methods*, SIAM Journal on Optimization **13** (2002).

[Geo72] A.M. Geoffrion, *Generalized Benders decomposition*, Journal of Optimization Theory and Applications **10** (1972), no. 4, 237–260.

[Hin01] M. Hintermüller, *A proximal bundle method based on approximate subgradients*, Computational Optimization and Applications **20** (2001), 245–266, 10.1023/A:1011259017643.

[HUL93] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms*, Grund. der math. Wiss, no. 305-306, Springer-Verlag, 1993, (two volumes).

[Kel60] J. E. Kelley, *The cutting plane method for solving convex programs*, J. Soc. Indust. Appl. Math. **8** (1960), 703–712.

[Kiw06] K. C. Kiwiel, *A proximal bundle method with approximate subgradient linearizations*, SIAM Journal on Optimization **16** (2006), no. 4, 1007–1023.

[Lem01] C. Lemaréchal, *Lagrangian relaxation*, Computational Combinatorial Optimization (M. Jünger and D. Naddef, eds.), Springer Verlag, Heidelberg, 2001, pp. 112–156.

[LNN95] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov, *New variants of bundle methods*, Math. Program. **69** (1995), no. 1, 111–147.

[OS13] W. Oliveira and C. Sagastizábal, *Level bundle methods for oracles with on-demand accuracy*, Tech. Report Submitted for publication, preprint available on optimization-online `http://www.optimization-online.org/DBHTML/2012/03/3390.html`, 2013.

[OSL13] W. Oliveira, C. Sagastizábal, and C. Lemaréchal, *Bundle methods in depth: a unified analysis for inexact oracles*, Tech. Report Submitted for publication, preprint available on optimization-online `http://www.optimization-online.org/DBHTML/2013/02/3792.html`, 2013.

[OSS11] W. Oliveira, C. Sagastizbal, and S. Scheimberg, *Inexact bundle methods for two-stage stochastic programming*, SIAM Journal on Optimization **21** (2011), no. 2, 517–544.

[SDR09] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: Modeling and theory*, MPS-SIAM Series on Optimization, SIAM - Society for Industrial and Applied Mathematics and Mathematical Programming Society, Philadelphia, 2009.

[Sol03] M.V. Solodov, *On approximations with finite precision in bundle methods for nonsmooth optimization*, Journal of Optimization Theory and Applications **119** (2003), no. 1, 151–165.

[ZPR00] G. Zakeri, A. Philpott, and D. Ryan, *Inexact cuts in benders decomposition*, SIAM Journal on Optimization **10** (2000), no. 3, 643–657.